John von Neumann Institute for Computing

Parallel Endmember Extraction Techniques Applied to a Self-Organizing Neural Network for Hyperspectral Image Classification

D. Valencia, A. Plaza, R.M. Pérez, M.C. Cantero,
P. Martínez, J. Plaza

published in

Parallel Computing:
Current & Future Issues of High-End Computing,
Proceedings of the International Conference ParCo 2005,
G.R. Joubert, W.E. Nagel, F.J. Peters, O. Plata, P. Tirado, E. Zapata (Editors),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **33**, ISBN 3-00-017352-8, pp. 591-598, 2006.

http://www.fz-juelich.de/nic-series/volume33

Parallel Endmember Extraction Techniques Applied to a Self-Organizing Neural Network for Hyperspectral Image Classification

D. Valencia*[a] A. Plaza[a], R.M. Pérez[a], M.C. Cantero[a], P. Martínez[a], J. Plaza[a]

[a]Neural Networks and Signal Processing Group (GRNPS), Computer Science Department,
Computer Architecture and Technology Section Avda. de la Universidad s/n,
University of Extremadura, E-10071 Cáceres, Spain

1. abstract

Advances in remote sensing technology have recently led to the development of hyperspectral sensor instruments, capable of collecting hundreds of images for the same area on the surface of the Earth at different wavelengths in the electromagnetic spectrum. Such images, which can be considered spatially and spectrally continuous, are characterized by their extremely large dimensionality. The identification of pure spectral constituents (called *endmembers*) in those images is considered to be a crucial task in hyperspectral data exploitation. Endmember extraction algorithms are computationally very expensive, due to the fact that they are based on complex mathematical operations. However, both the intrinsic properties of the image data and regularities in computations make these algorithms suitable for parallel implementation. In order to exploit the proposed techniques in applications that require a response in near real time, this paper investigates parallel implementations of a combined morphological/neural classification algorithm for hyperspectral imagery. The proposed implementation has been developed by considering two possible data-domain partitioning schemes: spatial-domain parallelism and spectral-domain parallelism. The performance of the parallel algorithm is tested on Thunderhead, a 256-processor massively parallel Beowulf cluster at NASAs Goddard Space Flight Center in Maryland.

2. Introduction

Last generation hyperspectral sensors have recently demonstrated their potential in land cover identification and characterization applications [1]. Each pixel collected by those sensors is given by a high-dimensional vector of values that provides a "spectral signature", which can be used to accurately characterize the composition of each site. A common technique for hyperspectral image classification relies on the identification of pure spectral signatures, often called spectral "endmembers" due to the fact that they are typically located on the corners of the hyperdimensional cube defined by the data volume. These pure signatures can be used to produce both *hard* and *soft* classifications, where a pixel vector may be classified into a single pure class or several mixed classes with different sub-pixel proportions.

One of the most successful approaches to endmember extraction in the literature has been the Automated Morphological Endmember Extraction (AMEE) algorithm [2], which is fully automated and does not require any data pre-processing. This technique successfully integrates both the spatial and spectral information in the data to conduct a multidimensional endmember search. The algorithm is computationally expensive due to complex mathematical operations involved. However, both the intrinsic properties of the image data and the regularities in endmember-based computations make

*Corresponding author. Contact E-mail: davaleco@unex.es

this algorithm highly amenable to parallel implementation. It should also be noted that the endmember pixels provided by endmember extraction algorithms such as AMEE are suitable to be used as input information for other applications. For instance, there are many situations where a detailed knowledge of image endmembers is not sufficient to extract a detailed land-cover classification map. In this context, artificial neural networks (ANNs) have demonstrated to be a powerful tool for land-cover classification because the information provided by ANNs can not only be used to provide *hard* labels, but also to obtain *soft* classification labels, e.g., by taking into account the degree of membership or similarity of a certain input pattern (pixel vector) to a certain output class (endmember). In the field of ANN-based remotely sensed data interpretation, Kohonen's self-organizing map (SOM) has been widely recognized as a very powerful tool to perform both hard and soft classification. This model is based on an unsupervised learning strategy that does not require any previous test samples [3,4]. Again, one of the main restrictions of SOM-based analysis is its high computational complexity, which is a serious drawback in applications that require a response in near real-time, such as those aimed at detecting and/or tracking natural disasters such as forest fires, oil spills, and other types of chemical contamination.

In this paper, we develop a parallel implementation of a combined AMEE/SOM (morphological/neural) approach to hyperspectral image classification. Although a few parallel algorithms for hyperspectral imaging exist in the literature [5], our parallel approach is the first one that integrates both spatial and spectral information in simultaneous fashion. It relies on data-parallel domain decomposition techniques aimed at minimizing inter-processor communication and maximizing load balance. It should be noted that the proposed method relies on well-known strategies, which have been widely used in the past for handling parallel computations in other research areas including High-Performance Fortran (HPF) and parallel skeletons [6,7]. However, the application of the proposed data-parallel strategy to hyperspectral imaging is new and represents a novel contribution. The paper is structured as follows. Section 3 describes the algorithm proposed for parallelization. Section 4 discusses key features related to the parallelization of the algorithms and their implementation. Section 5 reports parallel performance and classification results achieved by our combined AMEE/SOM approach. Finally, section 6 concludes with some remarks and hints at plausible future research.

3. Algorithms

In this section, we briefly address the fundamental properties of the individual AMEE and SOM techniques. These methods are available in the open literature and we will not expand on their detailed properties here, but relevant hints for their parallelization will be pointed out.

3.1. Automated morphological endmember extraction (AMEE)

In order to define extended morphological operations in hyperspectral imaging, we first impose an ordering relation in terms of spectral purity in a set of neighboring pixel vectors lying within a kernel neighborhood, known as structuring element (SE) in mathematical morphology terminology [2]. Using the two basic morphological operations illustrated in Fig. 1, the AMEE algorithm calculates a *morphological eccentricity index*(MEI) by comparing the output of the dilation to the output of the erosion for each pixel in the input data, using the SE in sliding-window fashion [2]. The MEI index is updated by repeating the above procedure by several algorithm iterations, where the result of the morphological dilation replaces the input data at the end of each iteration. The complexity of AMEE algorithm is $O(p_f x p_B x I_{MAX} x N)$, where p_f is the number of pixels of the input hyperspectral image f; p_B is the number of pixels in the structuring element; I is the number of iterations executed by

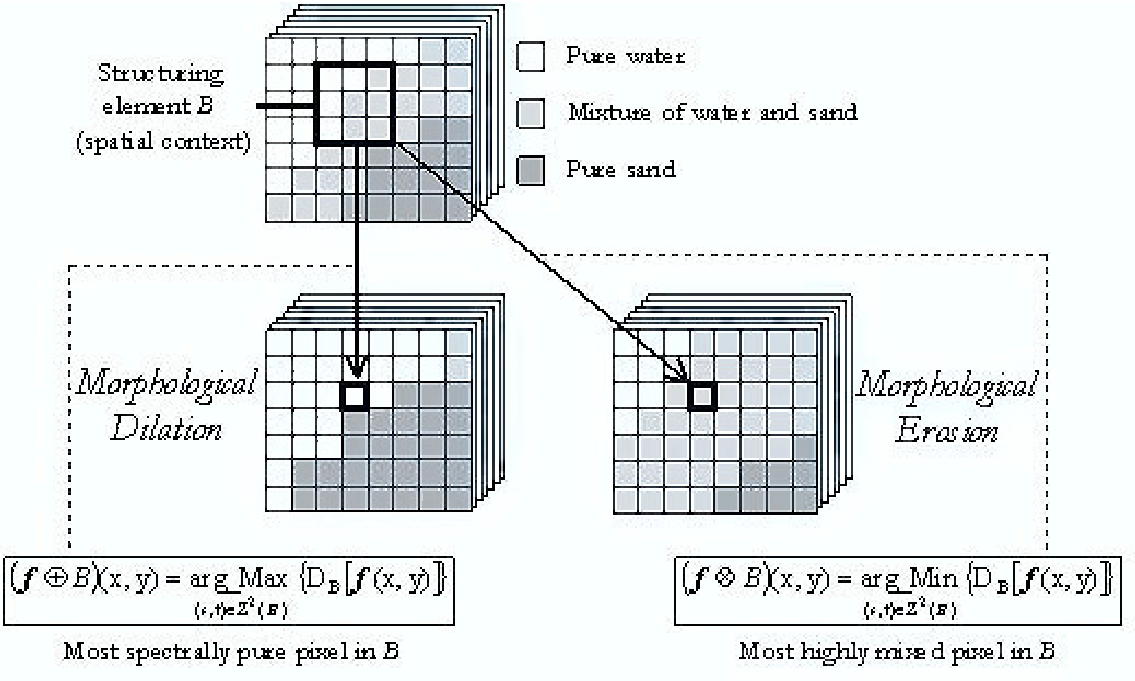

Figure 1. Morphological operations extended to hyperspectral imagery.

the algorithm; and N is the number of spectral bands. This results in very high computational complexity, in particular, when the value of N is very large [2]. Parallelization of AMEE must take into account the data dependencies introduced by the adopted sliding window-based approach, illustrated in Fig. 1.

3.2. Self-organizing map (SOM)

The neural model proposed in this work consists of N input neurons and M output neurons [3], where N is the dimensionality of the input pixel vectors, and M is the number of endmembers or class prototypes provided by AMEE algorithm. The network consists of two layers, with feedforward connections from the input layer to the output layer and a set of associated connection weights, arranged in a matrix that will be denoted hereinafter as W_{MxN}. The working procedure of the network is given by two different stages: clustering and training. In the former, the endmembers found by AMEE are presented to the network so that feedforward connections change and adapt to the information provided by the spectral data. In the training stage, feedforward connections project input patterns onto the feature space, and the Euclidean distance is used to identify a winning neuron. This procedure is summarized below:

1. Weight initialization. Normalized random values are used to initialize the weigth vectors: $w_i^{(0)}$, with $i = 1, 2, ..., M$.

2. Training. In this work, this step is accomplished by using AMEE-generated endmember signatures.

3. Clustering. For each input pattern x, a winning neuron i^* is obtained at time t by using an Euclidean distance-based similarity criterion, i.e., $i^*[x] = \min_{1<=j<=M} ||x - w_j||^2$.

4. Weight adjustment. The winning neuron (and those neurons in the neighborhood of the winning

one) adapt their weights using the following expression, where $\alpha(t)$ and $\sigma(t)$ are the learning and neighbouring functions, respectively. $w_i^{(t+1)} = w_i^t + \sum_{t'=t_0}^{t_{max}} \alpha(t')\sigma(t')(x - w_i^{(t)})$

5. Stopping rule. The SOM algorithm terminates as soon as a pre-determined number of iterations, t_{max}, has been accomplished.

4. Parallelization

Two types of data parallelism can be exploited in the proposed algorithm: spatial-domain parallelism and spectral-domain parallelism. Spatial-domain parallelism subdivides the input image into multiple blocks made up of entire pixel vectors, and assigns one or more blocks to each processing element (PE). On other hand, the spectral-domain parallel paradigm subdivides the whole multiband data into blocks made up of contiguous spectral bands (sub-volumes), and assigns one or more sub-volumes to each PE. In the following, we provide a discussion on the two types of parallelism above and their impact on the individual steps (morphological/neural) of the proposed method.

4.1. Parallelization of the morphological algorithm

In previous work, we have analyzed several parallelization strategies for the morphological stage of our combined algorithm [8,9]. Those studies revealed that, when the input data partitioning is accomplished in the spectral domain, the local spatial information of the images remains together. This is an important property for parallelization of image processing algorithms of the window-moving type, because this paradigm allows division of the input data into sub-volumes that can be processed independently by the AMEE algorithm. Also, this approach does not need to hold any replicated information to complete the calculations involved in the morphological process. Despite the above remarks, our previous work [9] has demonstrated that the spectral domain decomposition paradigm is not suitable (in general) for hyperspectral imaging applications. The main reason is that most hyperspectral imaging techniques consider the spectral information contained in each pixel vector as a unique entity. In other words, a spectral domain decomposition paradigm would break the spectral identity of the data because each pixel vector would be split amongst several PEs. If we take into account the fundamental characteristics of the morphological algorithm, the selection of a partitioning scheme in the spatial domain is critical for the success of our parallelization.

Several reasons justify our decision to implement a spatial domain-based partitioning framework. First, this strategy retains the spatial/spectral information, a desired property in a combined approach such as the one implemented by the proposed morphological algorithm. Since the resulting partitions are composed of spatially adjacent pixel vectors, the application of a sliding window-based approach can be accomplished in parallel with minimum changes to the original algorithm, thus enhancing code reusability and portability. A second reason has to do with the cost of inter-processor communication. In spectral-domain parallelism, the SE-based calculations made for each hyperspectral pixel need to originate from several PEs, and thus require intensive communications. A final major reason is that spatial information is particularly relevant in the local neighborhood around each pixel. Subsequently, partitioning in the spatial domain guarantees that spatial/spectral information can be retained at no extra cost by the proposed parallelization strategy [8]. To conclude this subsection, we emphasize that the main drawback of the proposed parallelization strategy for the morphological algorithm is the need to replicate information in order to reduce inter-processor communication [9]. However, we have experimentally proved that the cost of processing redundant information is insignificant compared to the cost of transmitting the boundary data. The introduction of replicated data introduces border-handling and overlapping issues which are simply resolved by

considering those pixels inside the local processor domain only for the MEI calculation [8].

4.2. Parallelization of the neural algorithm

In order to parallelize the SOM algorithm, we face similar problems than those already raised in the previous subsection. A straightforward approach to parallelization of the neural algorithm is to simply replicate the whole neural network architecture, which is a feasible approach due to the random nature of the initial weights of the network. However, this option would result in the need for very complex rules of reduction, and also in integrity hazards [4]. Taking into account our previous studies [4,8] and considering the relatively small size of the training set, we have experimentally tested that the overhead usually takes place in the training stage (i.e, in the form of Euclidean distance calculations and adjustment of weight factors). This fact makes partitioning of the weight matrix W_{MxN} a very appealing solution to reduce the computation time. Again, two main alternatives can be adopted to carry out such partitioning: (1) Division by input neurons (endmembers/training patterns); or (2) Division by output neurons (class prototypes). It should be noted that, in the latter case, the parallelization strategy is very simple. Quite opposite, when the former approach is adopted, there is a need to communicate both the calculations and the intermediate results among different processors. This introduces an overhead in communications that may significantly slow down the algorithm; according to our preliminary experiments, this option could even give worst results than those found by the sequential version of the SOM algorithm. On the other hand, the partitioning scheme based on dividing by class prototypes only introduces a minor communication overhead. This approach creates the need to introduce a broadcast/all-reduce protocol in order to obtain the class prototype through local minimum calculations, in batch-mode processing fashion. The winner neuron for each pattern needs to be tallied, and subsequent modifications for the weight update factor also need to be stored for further addition/subtraction. This approach also allows us to directly obtain the winner neuron at each iteration without the need for any further calculations. It also facilitates a pleasingly parallel solution which takes full advantage of the processing power available in the considered parallel architecture while, at the same time, minimizing the overhead introduced by inter-processor communications.

At this point, we must emphasize that the proposed parallel scheme still creates the need to replicate calculations to further reduce communications. However, the amount of replicated data is limited to the complete training pattern set, which is stored at every local processor along with administrative information, such as the processor that holds the winner neuron, the processor that holds neurons in the neighborhood of the winner neuron, etc. Such information can also be used to reduce the communication overhead even further. For instance, we have considered two different implementations of the neighborhood modification function $\sigma(t')$: the first one is applied when a node is *inside* the neighborhood of the winner neuron, while the second is used when the node is *outside* the domain of that processor. To assess integrity of the considered function, a look-up table is locally created at each processor to tally the value of $\sigma(t)$ for every neuron pair. While in the present work the neighborhood function is gaussian, i.e., $\sigma(t) = e^{-\frac{\left|i^* - i\right|}{t}}$, other functions may also be considered as well [4]. In any regard, it is important to emphasize that when the neighborhood function is applied to the processor that holds the winner neuron, the neighborhood function is used in its traditional form. Quite opposite, when the function is applied to other processors, a modified version is implemented to average the distances to all possible winner neurons. This approach reduces the amount of communications and represents a more meaningful and robust neighborhood function [4]. As a final major remark, our MPI-based implementation makes use of blocking primitives to ensure that all processors are synchronized. This prevents integrity problems in the calculations related with

W_{MxN}.

5. Experimental results

The proposed parallel algorithm has been implemented in the C++ programming language using calls to message passing interface (MPI), where the MPICH 1.2.6 version was used in experiments due the demonstrated flexibility of this version to migrate the code to different platforms. The parallel algorithm has been tested on Thunderhead, a massively parallel Beowulf cluster at NASAs Goddard Space Flight Center in Maryland, where our parallel code is currently being exploited in various Earth-based remote sensing applications. Thunderhead is composed of 256 dual 2.4 Ghz Intel Xeon nodes, each with 1 Gb of memory and 80 Gb of main memory. The total peak performance of the system is 2457.6 Gflops. Before discussing the parallel performance achieved by the proposed algorithm, we briefly describe a hyperspectral scene (designated by AVIP92) that will be used for validation purposes in this work. The scene was collected by the NASA/JPL AVIRIS system [1] over a small area (145 lines by 145 samples) over the Indian Pines agricultural test site in Northwestern Indiana (available online from http://dynamo.ecn.purdue.edu along with 16 mutually exclusive ground-truth classes). Although the scene represents a challenging classification problem, the proposed algorithm achieved 90% overall accuracy and high individual test accuracies when applied to this scene. Fig. 2 shows the parallel performance of the morphological and neural algorithms, displayed separately for clarity. The two considered performance measures are the speedup and the parallel efficiency. In order to compute the speedup, we approximate the time required to complete a task on N parallel processors using $T(N) = A_N + \frac{B_N}{K}$ where A_N is the sequential (non-parallelizable) portion of the computation and B_N is the parallel portion. In the morphological algorithm, A_N is given by the sequence of operations implemented by the partitioning module, and B_N refers to the endmember extraction procedure. In the neural algorithm, A_N corresponds to the generation of random weight values, while B_N is dominated by the training procedure. We can define the speedup for N processors as $S(N) = \frac{T(1)}{T(N)} \approx \frac{A_N+B_N}{A_N+(\frac{B_N}{N})}$ where $T(1)$ denotes single processor time. Using the definitions above, we can further compute the parallel processing efficiency, defined as the actual speedup divided by the number of processors, i.e., $E_N = \frac{S_N}{N}$. As shown by Fig. 2, the

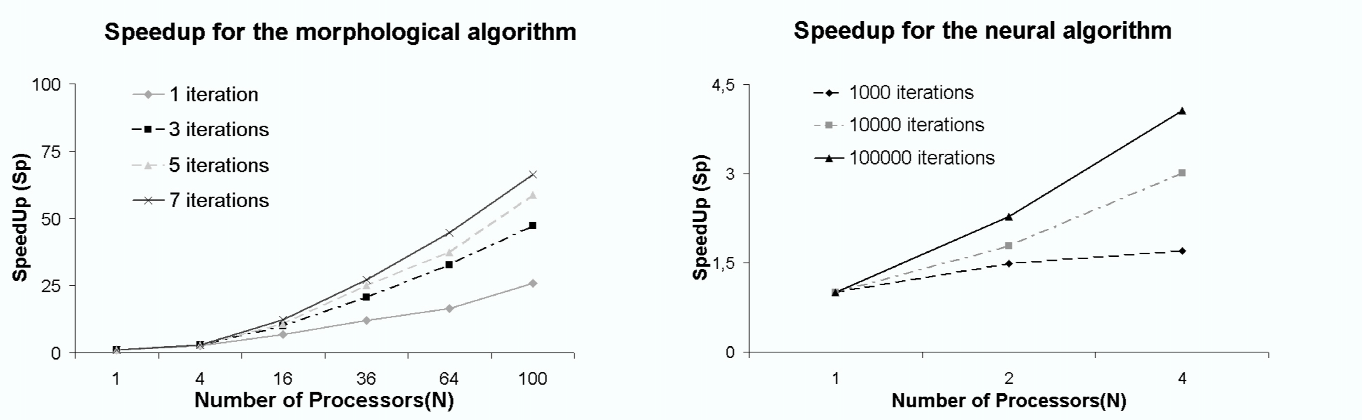

Figure 2. Speedup/parallel efficiency achieved by morphological/neural algorithms on Thunderhead.

morphological algorithm scales reasonably well on Thunderhead. This is because it takes advantage of some of the intrinsic characteristics of window-moving image processing algorithms, such as spatial and temporal data locality that result in cache reuse. The best speedup compromise for $I = 7$ algorithm iterations was achieved for $N = 16$ processors, with $S_{16} = 12.33$ and $E_{16} = 0.77$. The degradation in parallel efficiency as the number of processors is increased is likely due to the effect of redundant computations. On the other hand, Fig. 2 also reveals that, although parallelization of the neural algorithm is more complicated *a priori* due the expected impact of communications, the parallel neural code also scales relatively well (for a reduced number of processors). It should be noted that the cost of communications in the parallel neural algorithm cannot be reduced by introducing redundant computations, as it was the case in the morphological algorithm. Even though the amount of data to be exchanged is minimized by the proposed parallel neural strategy, we still had to deal with the size of the minimum transfer unit (MTU) of the communication network, a parameter that is not easily adjustable in the Thunderhead system. In future developments, we are planning on incorporating techniques able to automatically adjust the size of the MTU according to the properties of the input data. For instance, the domain of a single batch-mode iteration could be expanded to several network epochs (with all training patterns involved at each one) instead of just one epoch as in the current implementation. This could lead to much better data compaction inside the considered MTU. Also, we have detected that the parallel efficiency achieved for large training sets is significantly higher than that found for smaller training sets. This is because computations clearly dominate communications in this case, thus greatly enhancing the granularity of the parallel computation. As one would expect, the use of large training sets also results in much higher classification accuracies by the SOM neural network.

Although only results with 4 processors are reported in this work, we also observed that increasing the number of processors introduced fluctuations in the achieved speedups with significant drops in parallel efficiency. This is due in part to the scheduling policies implemented in the Thunderhead cluster, which tend to assign high priority to jobs that require a very large number of processors. Even in spite of the above limitations, our measured speedups reveal slight superlinear scaling effects in some cases, probably due to cache reuse (e.g., when 10^5 training iterations were considered, values of $E_2 = 1.135$ and $E_4 = 1.01$ were measured). This reveals that cache spatial and temporal locality could be partially used to overcome the limitations imposed by excessive communications. The above results also lead us to believe that the best configuration for the parallel SOM algorithm is likely to be achieved when most neural network partitions fit completely in the local processor caches. Further experimentation, however, is highly desirable in order to adapt the parallel properties of the neural algorithm to those observed in the morphological algorithm. In particular, there is a need to balance the combined computing power achieved by the pool of processors employed by the morphological algorithm and those used by the neural algorithm in the same algorithm run. This feature brings out new exciting future perspectives, such as the possibility to launch multiple neural-based classifiers in parallel. Such multiple classifier-based processing framework represents a completely novel data analysis paradigm in hyperspectral imaging, and previously looked too computationally complex to be developed in practical applications.

6. Conclusions

The aim of this paper has been the parallel implementation on high performance computers of an innovative morphological/neural technique for unsupervised classification of hyperspectral data sets. We show that parallel computing at the massively parallel level, supported by message passing, provides a unique framework to accomplish the above goal. For this purpose, computing systems

made up of arrays of commercial off-the-shelf computing hardware are a cost-effective way of exploiting this sort of parallelism in remote sensing applications. The two discussed parallelization strategies (morphological/neural) provide several intriguing findings and parallel design considerations that may help hyperspectral image analysts in selection of efficient algorithms for specific applications. Further, the proposed parallel strategies offer an unprecedented opportunity to explore methodologies in fields that previously looked to be too computationally intensive in practice, due to the inmense files common to remote sensing problems. The combination of this readily available computational power and the new perspectives introduced last generation sensor instruments may introduce major changes in the systems currently used by NASA and other agencies to process Earth and planetary remotely sensed data.

References

[1] R.O. Green et al.:"Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS)". *Remote Sensing of Environment*, vol. 65, pp. 227-248. 1998.

[2] A. Plaza, P. Martinez, R. Perez and J. Plaza.: "Spatial/spectral endmember extraction by multidimensional morphological operations". *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 9, pp. 2025-2041, Sept. 2002.

[3] T. Kohonen: "Self-Organizing Maps". *Springer*, Berlin, Heidelberg, 1995. (Second Edition 1997).

[4] P. Martinez, P.L. Aguilar, R. Perez and A. Plaza.: "Systolic SOM Neural Network for Hyperspectral Image Classification" in: *Neural Networks and Systolic Array Design*. Edited by D. Zhang and S.K. Pal. World Scientific, pp. 26-43, 2002.

[5] T. Achalakul and S. Taylor: "A distributed spectral-screening PCT algorithm". *Journal of Parallel and Distributed Computing*, vol. 63, pp. 373-384, 2003.

[6] M. Chandy, I. Foster, K. Kennedy, C. Koelbel, and C-W. Tseng.: "Integrated Support for Task and Data Parallelism". *International Journal of Supercomputer Applications*, vol. 8, pp. 80-98, 1994.

[7] M. Cole. Skeletal Parallelism home page. Available online: http://homepages.inf.ed.ac.uk/mic/Skeletons/

[8] A. Plaza, D. Valencia, P. Martinez and J. Plaza.: "Parallel Implementation of Algorithms for Endmember Extraction from AVIRIS Hyperspectral Imagery". *XIII NASA/Jet Propulsion Laboratory Airborne Earth Science Workshop*. Pasadena, CA, USA, 2004.

[9] D. Valencia, A. Plaza, P. Martinez and J. Plaza.: "On the Use of Cluster Computing Architectures for Implementation of Hyperspectral Analysis Algorithms". *Proceedings of the 10th IEEE Symposium on Computers and Communications*, pp. 995-1000, Cartagena, Spain, 2005.

[10] J. Dorband, J. Palencia and U. Ranawake.: "Commodity computing clusters at Goddard Space Flight Center". *Journal of Space Communication*, vol. 1, no. 3, pp. 60-75, 2003.